Formal Knowledge Management in Distributed Environments

Formal Knowledge Management in Distributed Environments*

W. Marco Schorlemmer[1], Stephen Potter[1], David Robertson[1], and Derek Sleeman[2]

[1] CISA, Division of Informatics, The University of Edinburgh
[2] Department of Computing Science, University of Aberdeen

In order to address problems stemming from the dynamic nature of distributed systems, there is a need to be able to express notions of evolution and change of knowledge components of such systems. This need becomes more pressing when one considers the potential of the Internet for distributed knowledge-based problem solving — and the pragmatic issues surrounding knowledge integrity and trust this raises. We introduce a formal calculus for describing transformations in the 'lifecycles' of knowledge components, along with suggestions about the nature of distributed environments in which the notions underpinning the calculus can be realised. The formality and level of abstraction of this language encourages the analysis of knowledge histories and allows useful properties about this knowledge to be inferred.

Formal Lifecycles in a Brokering Architecture

We take the real-life example of Ecolingua[3], an ontology for ecological meta-data. It was constructed on the Ontolingua Server[4] by reusing classes from other ontologies in the server's library, and then automatically translated into Prolog syntax by the server's translation service.

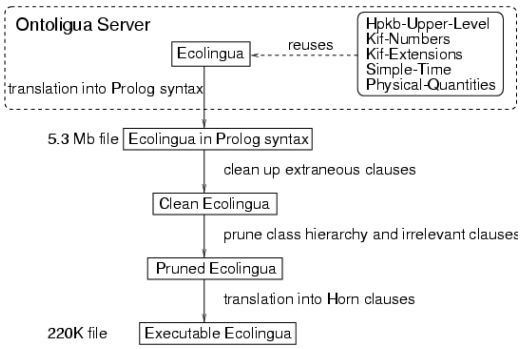

Fig. 1. Ecolingua's lifecycle

Because the outcome of the translation process was an overly large 5.3 Mb file, and in order to get a smaller and more manageable set of axioms, it was necessary to reduce the ontology, by implementing filters that first deleted all extraneous clauses, and then pruned the class hierarchy and removed irrelevant clauses accordingly. Finally, a translation into Horn clauses was performed in order to use the ontology with a Prolog interpreter (see Figure 1).

We postulate that particular sequences of lifecycle steps like those of Figure 1 might be common in particular domains and perhaps with particular forms of knowledge component. The ability to generalise and 'compile' these sequences into *lifecycle patterns* would encourage more efficient behaviour when faced with the need to make similar modifications in the future.

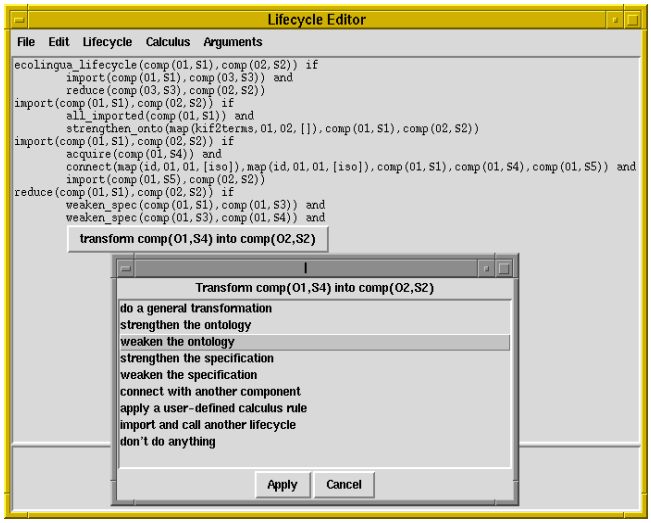

Fig. 2. Editing Ecolingua's lifecycle

For this reason we have implemented a *lifecycle editor and interpreter* that (1) enables a knowledge engineer to analyse the lifecycle of a knowledge component, extract its abstract pattern, and devise a formal representation of it (see Figure 2), using rules of a formal lifecycle calculus that capture abstract lifecycle steps such as *weaken the ontology* by means of a channel-theoretic semantics[5]; and (2) is capable of executing the formal representation of a lifecycle.

Since lifecycle patterns are described at a generic level, we have implemented a *brokering service* that enacts the lifecycle execution in a distributed environment, so that a knowledge engineer can choose among several solvers capable of performing abstract lifecycle steps in a particular domain-specific fashion.

The lifecycle execution generates a *lifecycle history* of the knowledge component. This lifecycle history can later be used to infer properties of the components, not by inspecting the specification of the component itself — since this would require the cumbersome task of reasoning with the axioms that constitute the specification — but by inspecting the structure of its lifecycle.

* Supported under the AKT IRC, which is sponsored by the UK EPSRC under grant GR/N15764/01, and comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

[3] Corrêa da Sliva et al.: On the insufficiency of ontologies: problems in knowledge sharing and alternative solutions. Knowledge-Based Systems **15** (2002) 147–167.

[4] Farquhar, A., Fikes, R., Rice, J.: The Ontolingua Server: a tool for collaborative ontology construction. International Journal of Human-Computer Studies **46** (1997) 707–727.

[5] Barwise, J., Seligman, J.: Information Flow: The Logic of Distributed Systems. Cambridge University Press (1997).